using Microsoft.Kinect;

using System;

using System.Collections.Generic;

using System.Diagnostics;

using System.IO;

using System.Linq;

using System.Windows;

using System.Windows.Media;

using System.Windows.Shapes;

namespace Capstone

{

/// <summary>

/// Interaction logic for MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

#region Member Variables

private KinectSensor \_KinectDevice;

private readonly Brush[] \_SkeletonBrushes;

private Skeleton[] \_FrameSkeletons;

#endregion Member Variables

#region Constructor

public MainWindow()

{

InitializeComponent();

\_SkeletonBrushes = new Brush[] { Brushes.Black, Brushes.Crimson, Brushes.Indigo, Brushes.DodgerBlue, Brushes.Purple, Brushes.Pink };

KinectSensor.KinectSensors.StatusChanged += KinectSensors\_StatusChanged;

this.KinectDevice = KinectSensor.KinectSensors.FirstOrDefault(x => x.Status == KinectStatus.Connected);

}

#endregion Constructor

#region Methods

private void KinectSensors\_StatusChanged(object sender, StatusChangedEventArgs e)

{

switch (e.Status)

{

case KinectStatus.Initializing:

case KinectStatus.Connected:

case KinectStatus.NotPowered:

case KinectStatus.NotReady:

case KinectStatus.DeviceNotGenuine:

this.KinectDevice = e.Sensor;

break;

case KinectStatus.Disconnected:

//TODO: Give the user feedback to plug-in a Kinect device.

this.KinectDevice = null;

break;

default:

//TODO: Show an error state

break;

}

}

// Listing 4-2

private void KinectDevice\_SkeletonFrameReady(object sender, SkeletonFrameReadyEventArgs e)

{

using (SkeletonFrame frame = e.OpenSkeletonFrame())

{

if (frame != null)

{

Polyline figure;

Brush userBrush;

Skeleton skeleton;

LayoutRoot.Children.Clear();

frame.CopySkeletonDataTo(this.\_FrameSkeletons);

Skeleton[] dataSet2 = new Skeleton[this.\_FrameSkeletons.Length];

frame.CopySkeletonDataTo(dataSet2);

for (int i = 0; i < this.\_FrameSkeletons.Length; i++)

{

skeleton = this.\_FrameSkeletons[i];

if (skeleton.TrackingState == SkeletonTrackingState.Tracked)

{

userBrush = this.\_SkeletonBrushes[i % this.\_SkeletonBrushes.Length];

//Draw head and torso

figure = CreateFigure(skeleton, userBrush, new[] { JointType.Head, JointType.ShoulderCenter, JointType.ShoulderLeft, JointType.Spine,

JointType.ShoulderRight, JointType.ShoulderCenter, JointType.HipCenter

});

LayoutRoot.Children.Add(figure);

figure = CreateFigure(skeleton, userBrush, new[] { JointType.HipLeft, JointType.HipRight });

LayoutRoot.Children.Add(figure);

//Debug.WriteLine("------------------------------" + skeleton.Joints[joints[0]]);

//Draw left leg

figure = CreateFigure(skeleton, userBrush, new[] { JointType.HipCenter, JointType.HipLeft, JointType.KneeLeft, JointType.AnkleLeft, JointType.FootLeft });

LayoutRoot.Children.Add(figure);

//Draw right leg

figure = CreateFigure(skeleton, userBrush, new[] { JointType.HipCenter, JointType.HipRight, JointType.KneeRight, JointType.AnkleRight, JointType.FootRight });

LayoutRoot.Children.Add(figure);

//Draw left arm

figure = CreateFigure(skeleton, userBrush, new[] { JointType.ShoulderLeft, JointType.ElbowLeft, JointType.WristLeft, JointType.HandLeft });

LayoutRoot.Children.Add(figure);

//Draw right arm

figure = CreateFigure(skeleton, userBrush, new[] { JointType.ShoulderRight, JointType.ElbowRight, JointType.WristRight, JointType.HandRight });

LayoutRoot.Children.Add(figure);

saveCoordinates(skeleton);

}

}

}

}

}

// Listing 4-3

private Polyline CreateFigure(Skeleton skeleton, Brush brush, JointType[] joints)

{

Polyline figure = new Polyline();

figure.StrokeThickness = 8;

figure.Stroke = brush;

for (int i = 0; i < joints.Length; i++)

{

figure.Points.Add(GetJointPoint(skeleton.Joints[joints[i]]));

}

return figure;

}

private Point GetJointPoint(Joint joint)

{

DepthImagePoint point = this.KinectDevice.MapSkeletonPointToDepth(joint.Position, this.KinectDevice.DepthStream.Format);

point.X \*= (int)this.LayoutRoot.ActualWidth / KinectDevice.DepthStream.FrameWidth;

point.Y \*= (int)this.LayoutRoot.ActualHeight / KinectDevice.DepthStream.FrameHeight;

return new Point(point.X, point.Y);

}

#endregion Methods

#region Properties

public KinectSensor KinectDevice

{

get { return this.\_KinectDevice; }

set

{

if (this.\_KinectDevice != value)

{

//Uninitialize

if (this.\_KinectDevice != null)

{

this.\_KinectDevice.Stop();

this.\_KinectDevice.SkeletonFrameReady -= KinectDevice\_SkeletonFrameReady;

this.\_KinectDevice.SkeletonStream.Disable();

this.\_FrameSkeletons = null;

}

this.\_KinectDevice = value;

//Initialize

if (this.\_KinectDevice != null)

{

if (this.\_KinectDevice.Status == KinectStatus.Connected)

{

this.\_KinectDevice.SkeletonStream.Enable();

this.\_FrameSkeletons = new Skeleton[this.\_KinectDevice.SkeletonStream.FrameSkeletonArrayLength];

this.KinectDevice.SkeletonFrameReady += KinectDevice\_SkeletonFrameReady;

this.\_KinectDevice.Start();

}

}

}

}

}

#endregion Properties

private void saveCoordinates(Skeleton skeleton)

{

string filePath = @"d:\vids\";

string \_fileName = string.Format("{0}{1}{2}", filePath, DateTime.Now.ToString("MMddyyyyHmmss"), ".txt");

StreamWriter coordinatesStream = new StreamWriter(\_fileName);

foreach (Joint joint in skeleton.Joints)

{

coordinatesStream.WriteLine(joint.JointType + ", " + joint.TrackingState + ", " + joint.Position.X + ", " + joint.Position.Y + ", " + joint.Position.Z);

}

coordinatesStream.Close();

}

//public void saveVideo()

//{

// string filePath = timeStamp + ".txt";

// StreamWriter cooStream = new StreamWriter(filePath, false);

// IReadOnlyDictionary<JointType, Joint> joints = body.Joints;

// Dictionary<JointType, Point> jointPoints = new Dictionary<JointType, Point>();

// foreach (JointType jointType in joints.Keys)

// {

// ColorSpacePoint depthSpacePoint = this.coordinateMapper.MapCameraPointToColorSpace(joints[jointType].Position);

// cooStream.WriteLine(joints[jointType].JointType + " " + joints[jointType].TrackingState + " " + joints[jointType].Position.X + " " + joints[jointType].Position.Y + " " + joints[jointType].Position.Z + " " + depthSpacePoint.X + " " + depthSpacePoint.Y);

// }

// string wrtLineData = "LeftHand " + body.HandLeftState + " RightHand " + body.HandRightState;

// cooStream.WriteLine(wrtLineData);

// cooStream.Close();

//}

}

}

/\*

foreach(Joint joint in body.Joints)

{

coordinatesStream.WriteLine(joint.JointType + ", " + joint.TrackingState + ", " + joint.Position.X + ", " + joint.Position.Y + ", ", joint.Position.Z);

}

coordinatesStream.Close();

\*/